Report I

Miguel Rodríguez, RA: 192744, Email: m.rodriguezs1990@gmail.com

I. Problem

The objective of this work is to perform some basic digital image processing

A. Image comparison

it's necessary calculate distance between two images, for this, must use the metric of euclidean distance between the normalized histograms of each image. This exercise must be carried out using different numbers of bins for the construction of the histogram eg. 4, 8, 32, 128, 256, etc.

B. Bit plane spliting and Entropy

This section is divided into two tasks:

1) Bit plane spliting: Extract the bit planes of a grayscale image, compare the images obtained and explain the results.

2) Entropy: From the images obtained in I-B1, calculate the entropy using eq. 1

$$H = -\sum_{n=0}^{L-1} p_i \ log_2 \ p_i \tag{1}$$

Where L is the number of intensities of the imagen I, and p_i is the probability of a pixel to assume that intensity, The probability calculation is given by eq. 2

$$p_i = \frac{n_i}{n} \tag{2}$$

Where n_i is the frequency of appearance of that pixel in the image I, and n is the number of pixels in the image.

II. Solution

A. Image comparison

In the world of image processing, computer vision and other areas that work with images, there is a very common problem. which is the comparison between images.This is a problem that does not have a definitive solution, since the solution to the problem will depend on many variables, such as the type of images to be compared, the problem to solve, the sensor that acquired the image, etc.

For this work we are going to try with a naive solution, which consists in making a comparison using euclidean distance between the normalized histograms of the images.

before explaining how the problem was solved, it is necessary to have certain clear concepts such as: **histogram**, **histogram normalization**, **algebraic distance**, **descriptor**.

1) **Histogram**: it's a graphical representation of the relative frequency of all values taken by pixels in an image I, these are grouped in bins, while the greater is number of bins better will be the histograms accuracy. In Listing 1 can see a simple implementation of the algorithm to obtain the histogram of an image.

```
def histogram(image):
 hist = np.zeros(256)
 height, width = image.shape

 for y in range(0, width):
  for x in range(0, height):
   hist[image[y][x]] = hist[image[y][x]] + 1
 return hist

```

Listing 1: Histogram function

2) **Histogram normalization**: To compare different histogram of images, must be very careful with amount of pixels that each image has, because if the images have different amounts, their histograms can't be compared, by caused of they have different possible max values. To solve this problem, we use normalization, which allows us to transform histograms to comparable space. This normalization is done by dividing each of the frequencies by $N \times M$, being N and M the height and width of image I. After normalization, all frequencies will have values that vary into range $[0-1]$, which allows to compare two images because both image have the same possible max value. The value of each normalized bin of histogram represent the probability of occurrence this bin in the image.
3) **Algebraic distance**: Is a metric that allows to know how different or similar are two vectors or matrices, there are many types of distances and each distance is used for different cases depending on the problem that needs to be solved. In this case we will use the Euclidean distance, which is given by the eq. 3, where P and Q are the vectorial representation of the normalized histograms of images.

$$d_2(P,Q) = \sqrt{\sum_{i}^{n}(p_i - q_i)^2} \tag{3}$$

4) **Descriptor**: Is a vector representative of features of an image I, which is the conversion of I to a vector space. The descriptors describe fundamental features of images, such as color, texture or movement. its construction depends on the problem to be solved and how it is to be solved. it can also be combination of different tools of the image processing, like histogram, edges, entropy, norms, etc. Descriptors can be divided depending on the features they cover, color, texture, shape, movement or location.

After having these clear concepts, we can proceed to proposed solution, the first action was obtaining of the normalized histogram of each image in Fig 1, for this we use the function $cv2.calcHist$ of OpenCV library [1], for each channel in the images, then proceed to normalize the histograms dividing by


(a) Baboon


(b) Peppers


(c) Monalisa


(d) My normal photo


(e) My angry photo

Fig. 1: Original images used in comparison.

$M \times N$. The implementation of this algorithm can be seen in $normalized\_histogram$ function in Listing 2, which return an array of histograms , where each histogram belongs to each channel of image I. The visual representations of the histogrmas can be seen in Figs. 2, 3, 4, 5 and 6

```
def normalized_histogram(image, nbins):
 if len(image.shape) == 3:
  height, width, channels = image.shape
 else:
  height, width = image.shape
  channels = 1

 histr = []
 for i in range(channels):
  hist = cv2.calcHist([image],[i],None,[nbins
   ],[0,256]).flatten()
  hist = hist/(height * width)
  histr.append(hist)

 return np.array(histr)

```

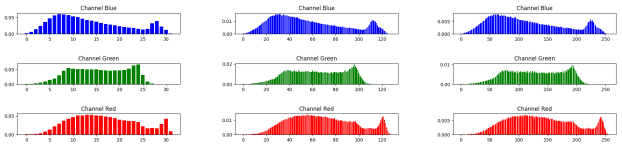
Listing 2: Histogram function

After obtaining histogram of images, we proceed to calculate the euclidean distance (eq. 3) between both images. in order to not implement this function, we use $np.linalg.norm$ function of NumPy [2].

The implementation can be seen in Listing 3, this function calculates distance between each histogram, and then calculates the average of the distances to obtain a total distance.

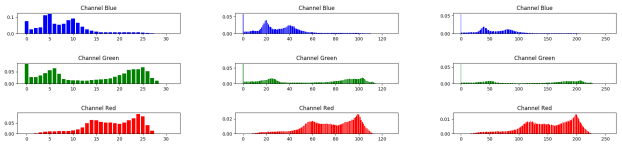
```
def euclidean_distance(hist1, hist2):
distance = 0.0

if(hist1.shape == hist2.shape):
for i in range(hist1.shape[0]):
distance = distance + np.linalg.norm(hist1[i] -
    hist2[i], 2)
distance = distance / hist1.shape[0]

return distance
```

Listing 3: Euclidean distance function



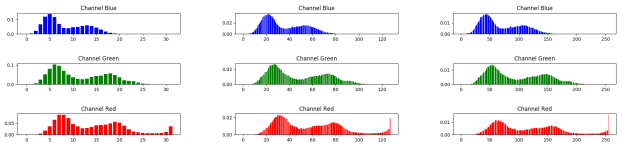
(a) 32 bins (b) 128 bins (c) 256 bins

Fig. 2: Histogram of baboon (1a) with different bins.



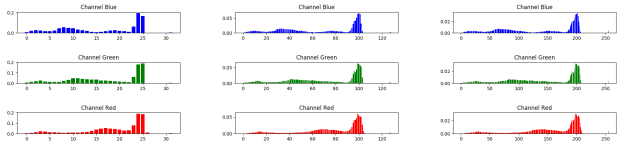
(a) 32 bins (b) 128 bins (c) 256 bins

Fig. 3: Histogram of peppers (1b) with different bins.



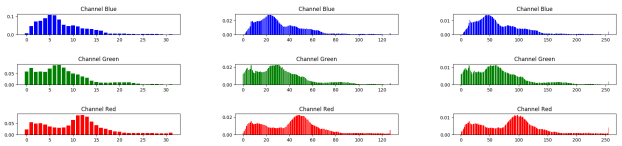
(a) 32 bins (b) 128 bins (c) 256 bins

Fig. 4: Histogram of monalisa (1c) with different bins.



(a) 32 bins (b) 128 bins (c) 256 bins

Fig. 5: Histogram of my normal photo (1d) with different bins.



(a) 32 bins (b) 128 bins (c) 256 bins

Fig. 6: Histogram of my angry photo (1e) with different bins.

	Baboon (1a)	Peppers (1b)	Monalisa (1c)	Normal (1d)	Angry (1e)
1a	0.0	0.15768	0.15581	0.24549	0.17594
1b	0.15768	0.0	0.18121	0.23447	0.17219
1c	0.15581	0.18121	0.0	0.31762	0.14493
1d	0.24549	0.23447	0.31762	0.0	0.31595
1e	0.17594	0.17219	0.14493	0.31595	0.0

TABLE I: Distance matrix using 32 bins

	Baboon (1a)	Peppers (1b)	Monalisa (1c)	Normal (1d)	Angry (1e)
1a	0.0	0.0887	0. 07956	0.13097	0.08883
1b	0.0887	0.0	0.10079	0.13128	0.10134
1c	0.07956	0.10079	0.0	0.16883	0.07410
1d	0.13097	0.13128	0.16883	0.0	0.16478
1e	0.08883	0.10134	0.07410	0.16478	0.0

TABLE II: Distance matrix using 128 bins

To prove efficiency of method, we performed different experiments, using all images shown in Fig 1, we calculated distance between all images, for this calculation we used three numbers of bins (32, 128 and 256). The result of these experiments can be seen in distances matrices shown in Tables I, II and III. Looking at these distance tables we can conclude: 1) While greater is number of bins smaller is distance between the images, because a large number of bins indicates that probabilistic distribution of these will be smaller, so the distance calculated by applying eq 3 will be much smaller because each of the bins will have smaller values. 2) Looking the results are shown in distance matrices (Table I, II and III), we can deduce that the closer to zero the value of calculated distance, more closer are images, so images like 1d and 1e, should give similar values, but seeing the results, are the ones that have greatest distantes between them, this can also be observed by seeing respective histograms in Figs. 5 and 6, which are very different. 3) Concluding from the experiments, measure euclidean distance between histograms of different images to compare, only allows to know how similar are their gray distribution, so it only serves to know how similar is an image to another through its tones, does not help to know if geometric shapes or appearance within the image are similar.

After performing the experiments, we have managed to conclude to know how similar are two image, we must use more robust descriptors than the histogram. Depending on the problem we want to solve is the shape of the descriptor.

B. Solution problem 2

This section is separated into two tasks: 1) Split an image in its eight binary layers. 2) Calculate entropy of each layer using eq 1 and conclude.

To perform the separation of image in its eight binary layers, it's necessary to undestand that a grayscale image can be represented as the mixture of eight image or binary layers, where each image fulfills a special role with respect to the information what he brings. each layer is associated with a special bit of the grayscale value of the image, the most significant image (8th) is an image where the pixel that are lit are those where the value in the grayscale has the 8th bit

	Baboon (1a)	Peppers (1b)	Monalisa (1c)	Normal (1d)	Angry (1e)
1a	0.0	0.06993	0.05670	0.09309	0.062937
1b	0.06993	0.0	0.07819	0.09786	0.07866
1c	0.05670	0.07819	0.0	0.11770	0.05266
1d	0.09309	0.09786	0.11770	0.0	0.11687
1e	0.062937	0.07866	0.05266	0.11687	0.0

TABLE III: Distance matrix using 256 bins

in one, the 7th image has pixels turned on where those pixels have in one the 7th bit of the chain, so on until 1st image, in which pixles on are those that have the 1st bit in one.

Each of these layers represents different levels of information of image, it is said that most significant information is in images where the bit that represent it is bigger (7th and 8th).

To perform the separation of images we use a function called $bit\_spliting$ shown in Listing 4, which return a list with the binary layers. To obtain these images we use the $bitwise\_and$ operator, and the powers of two from one to eight, with this it's possible to obtain each layer of the image I. The images used in the experiments are shown in Fig. 7.

```
def bit_spliting(image):
 images = []

 for i in range(8):
  n = 2**i
  new_image = np.bitwise_and(image,n)
  new_image[new_image == n] = 255
  images.append(new_image)

 return images
```

Listing 4: Bit spliting

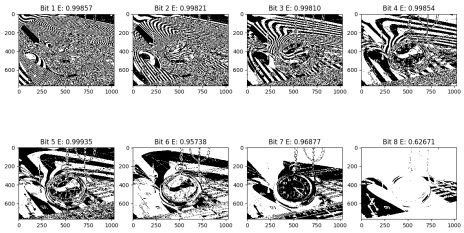
The image resulting from applying the Listing 4 in Figs. 7 can be seen in Fig. 8, which are sorted from the least significant bit to the most significant one. In these images, different patterns can be seen. The most significant layers are those that contain more information about the visual structure of the image, as in the case of the 7th and 8th layer. In the case of less significant layers, it can be seen that there is little or no visual information, which is still valuable information for the image.

In order to see how much information was provided by each of the layers we decided to perform two experiments in parallel, the stems comprised the reconstruction of the image starting from different paths.

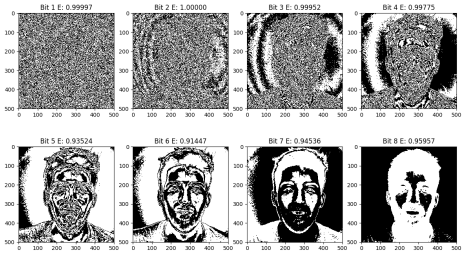
(a) Watch

(b) My normal photo

Fig. 7: Original images used in bit spliting

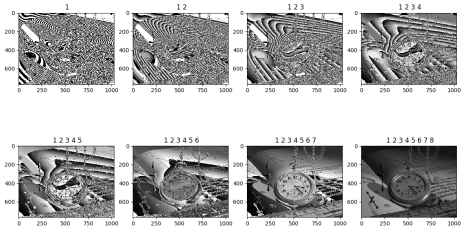


(a) Result of bit spliting Fig. 7a.

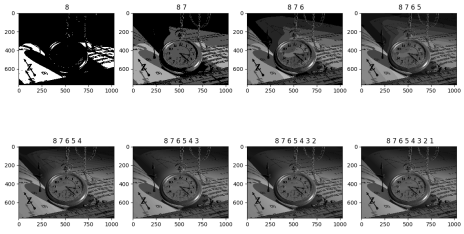


(b) Result of bit spliting Fig. 7b.

Fig. 8: Result of split images

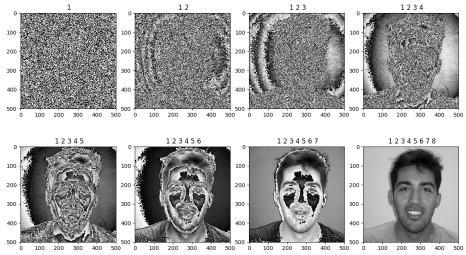


(a) Image reconstruction of Fig. 7a from least significant to most.

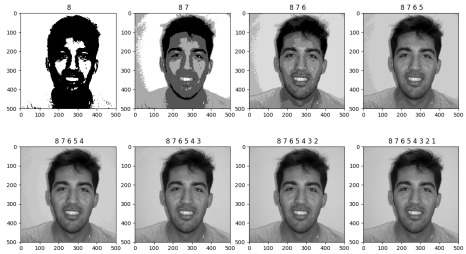


(b) Image reconstruction of Fig. 7a from most significant to least.

Fig. 9: Fig 7a reconstruction.



(a) Image reconstruction of Fig. 7b from least significant to most.



(b) Image reconstruction of Fig. 7b from most significant to least.

Fig. 10: Fig 7b reconstruction.

The first one consisted in reconstructing image from the least significant bit to the mos significant bit, as shown in Figs 9b and 10b. Here we can see that until 4th or 5th layer added, the relevant information of the image can't be seen or can't be understood for human eyes, instead after adding the most significant layers, the image makes sense and is easier undestand the exposed geometry.

In the second experiment, we reconstructed the image from the most significant bit at least significant, as shown in Fig. 9a and 10a. In this experiment we can see that adding the first two layers can obtain a very accurate image of the real image, in which only details of colors are missing, but the form and the most relevant visual information is already beign deployed. While adding more layers to the sum, the missing details are appearing in the image. Example, in case of Fig. 9a, we can seen that by adding first layers, the missing information represents shadows and image details, this can also be seen by looking Fig. 10a.

Reviewing the results obtained, we can conclude: 1) The most significant layers of an image have the most relevant information for taking decisions, so this images could be of great help when using then to build more robust and smaller descriptors.. 2) The least significant layers of an image are those that contain the details of the small variations in gray levels, they contain information relevant to shadows or color variations that are not very noticeable to the human eye. 3) This technique could be implemented in the compression of images,

because it is possible to reconstruct an image from the sum of all the layers, and also a binary image has a much smaller weight than a grayscale image.

After performing the separation of the binary layers of image we proceed to calculate the entropy of each of these layers, to make this calculation, we made use of the eq 1, which was implemented In the Listing 5. The calculation of the entropy is done on the histogram of an image, so we had to calculate the histogram for each layer, in this calculation we use the function implemented in the Listing 2.

```
def entropy(hist):
 entropy = 0
 width = hist.shape
 for x in range(0, width[0]):
  entropy = entropy + hist[x] * np.log2(hist[x])

 return -1 * entropy
```

Listing 5: Bit spliting

The results of the entropy calculation for each layer can be seen in Fig. 8, which are on each of the layers. Looking at each of the calculated entropies you can see that almost all entropies have values very close to 1. Considering that maximum entropy of a binary image is 1 and that the entropy of a system is maximal when all all events that can happen have the same probability, this implies that the distribution of zeros and ones is almost uniform in the binary layers. Therefore, according to the definition of entropy [3], while greater value of entropy more uncertainty one has,so more information is associated with the channel. In other words, if all possible values have the same probability, it means that it has a distribution close to uniformity, so each layer ontribute a significant amount of information. This informations does not have to be visual information or understandable to the human eyes.

With these results we can conclude: 1) The entropy measures the amount of information that passes through a channel, when the entropy is maximum, it means that much different information is being transported by that channel. 2) The entropy is maximum or minimum has no relation with the visual ability to understand the image. 3) The binary layers of an image contain vital information for the image, the most significant layers contain the information of the structure and the visual content, but the less significant layers contain the details with the colors and the small variations that exist between them.

REFERENCES

[1] G. Bradski, "Opencv library," *Dr. Dobb's Journal of Software Tools*, 2000.

[2] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011. [Online]. Available: http://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37

[3] H. Pedrini and W. Schwartz, *Análise de imagens digitais: princípios, algoritmos e aplicações*. THOMSON PIONEIRA, 2008. [Online]. Available: https://books.google.com.br/books?id=13KAPgAACAAJ

Miguel Rodriguez IEEE Member: 93224316, RA: 192744. MSc Student from UNICAMP, Brazil. Computers and Telecommunications Engineer from Diego Portales University, Chile. A charismatic young boy, empathic, a history and culture lover, motivated to travel around the world and learn about different cultures we can find in our planet.

The most noticed abilities are: the capacity to work as a team with people from different areas, the power to surpass problems without losing the desire and motivation to solve them, the great capacity and encouragement to learn new and interesting staffs, being this one, the best ability I earned during my college period.

Very interested in keep improving the skills in labor and the academic field; always searching new technologies and new techniques and moral challenges, especially in areas like artificial intelligence. A bike lover and technologies which use renewable energy, this due to the big motivation to build a future where technology and science can pacifically coexist with nature and that way contribute to improve the life quality in society.